ANNALS of Faculty Engineering Hunedoara – International Journal of Engineering. ISSN: 1584 - 2673

CONSIDERATIONS ABOUT NEURO-FUZZY ADAPTIVE SYSTEMS

Gelu Ovidiu TIRIAN

"Politechnica" University of Timisoara, Faculty of Engineering of Hunedoara,
Department of Electrical Engineering and Industrial Informatics, ROMANIA

Abstract:
This paper work describes the shaping up of a system based on a input-output data set using ANFIS. First of all, we had to design the input-output data set to use during the training. The next step is to design a FIS structure that is going to be used in order to shape up a system who must be appropriate to the input-output data set. The FIS structure is going to be used in order to establish a primary set of functions needed during the training process.
Keywords:
neuro-fuzzy system, shaping up, structure , use

1. INTRODUCTION

The neuronal networks [1] could add high dynamic performance to the classical methods of adjustment, if they are used and designed properly [7]. This is possible if they adjust in real time, during the adjustment process [4]. Due to their ability of approximating the non-linear arbitrary functions, artificial neuronal networks are widely used in the domain of non-linear systems. Artificial intelligence could be also used successfully for parameter identification or for estimating different features of the process. The most important properties of the artificial neuronal networks (which are important for different applications during the automatic management) are: the ability of function approximation. Due to this ability of approximating arbitrary non-linear functions, artificial neuronal networks are highly important for the non-linear systems; information spreading; artificial neuronal networks have a parallel structure that allows the implementation of the systems' hardware; such an approach enables a highly margin degree in case of deficiencies than the classical alternatives; a parallel calculation higher degree; the ability of adaptation; the ability of generalization; neuronal networks could be used based on the data available previously - „off-line" learning or „on-line" learning (in real time): if the network is correctly used then, it has the ability of turning the data to a general outline, which is not possible in case of the data-set in use; shaping up the multi-variable systems; genuinely the neuronal systems have several inputs and outputs, and they are used in multi-variable systems; high degree of sturdiness in case of noise and incomplete inputs.

2. MULTI-LAYER NEURONAL NETWORKS. TRAINING METHODS.

Multi-layer neuronal systems are highly important for the domain of the artificial intelligence, because a lot of applications and classification and acknowledgement problems such as, the approximation of non-linear functions, the control and identification of the non-linear systems or the adaptation percolation - are solved out by using different types of networks. Multi-layer neuronal networks [5] eliminate a lot of limited features of the classical methods of the simple propagation or multi-layer perceptron – we refer to XOR. Generally speaking, such networks use feed forward propagation – meaning that in the case of those networks, there is no reaction amongst the neurons within the hidden layers or amongst those within the output and input layers, like those in Figure no. 1. Training methods for such network belong to the supervised methods, and they develop alongside the error back propagation algorithm - ,,back propagation" [7]. The main characteristic of these networks is the active function – it has both active continuous and/or derived non-linear functions, such as in Figure no. 2.

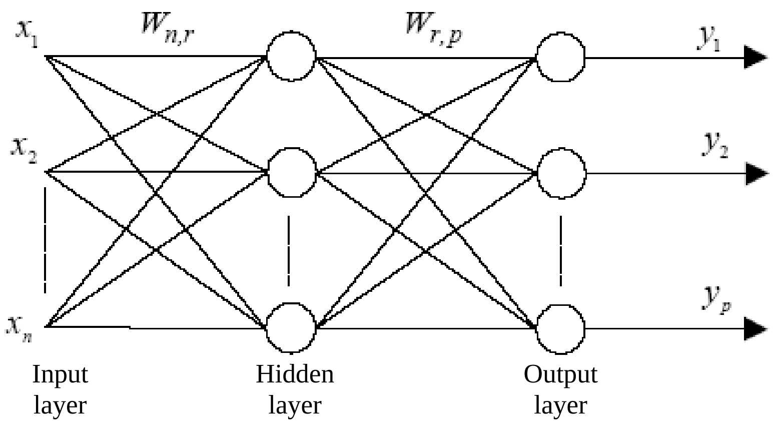


Figure no. 1 One hidden- layer neuronal network

The network in Figure no. 1 is made of one hidden layer who contains „n" neurons inside the input layer (n entries), „*r*" neurons inside the hidden layer, and „*p*" neurons inside the output layer [4]. The functions that make the neurons output layer active could be linear or sigmoid, meanwhile inside the hidden layer they must be sigmoid. In Figure no. 2 there are the most used functions that turn the multi-layer neuronal networks active, the sigmoid logarithmic function (left) and a hyperbole tangency function (right) are both coloured in blue, as well as their derivative functions - which are coloured in green.

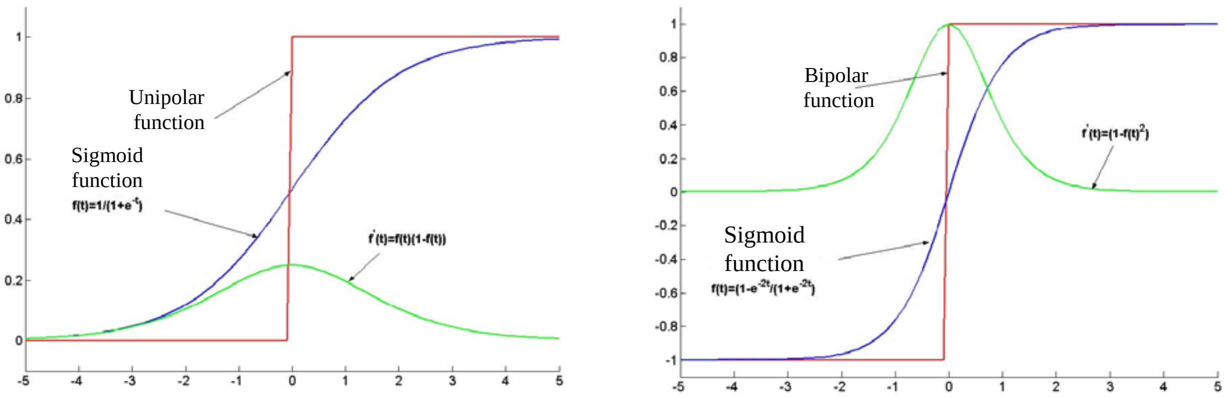


Figure no. 2 Functions that make multi-layer neuronal functions active

The error back propagation algorithm [7] uses three sets of data, one who turns the network active; other that is used for testing and validation of the weight we have come up with after the training stage; meanwhile the last one is the real data set. Choosing the data set for the training stage is highly important [29]. The data set must be representative for the matter under discussion, thus it should contain as much information as possible about the size-pairs of input values/output values we need. A correct data set will provide the network with the ability of generalization during the tests and the working process (real data sets), which reveal input values that have not been used during the training stage. Thus, the network shall come up with the right values at the output. We consider a training data set is made of „*m*" pairs who correspond to the most appropriate input and to the output vectors for the network:

$$\left(X^1,d^1\right),\left(X^2,d^2\right)\ldots\left(X^m,d^m\right) \quad (1)$$

In such conditions, when the input of the network uses the „*k*" data set, we could calculate the values for the output layer neurons, according to the moment when they turn active $s_j^k, j=1\ldots p$; as well as for the active function f^i, when the output value is the following:

$$y_j^k = f^i\left(s_j^k\right), \quad j=1\ldots p \quad (2)$$

We can define the square output error referring to the input vector X^k :

$$E_k = \frac{1}{2}\sum_{j=1}^{p}\left[d_j^k - y_j^k\right] = \frac{1}{2}\sum_{j=1}^{p}\left[d_j^k - f^i\left(s_j^k\right)\right] = \frac{1}{2}\left(e^k\right)^T e^k \qquad (3)$$

Total error for all „*m*" data-sets is the sum of the E_k errors:

$$E = \sum_{k=1}^{m} E_k \qquad (4)$$

3. NEURO-FUZZY ADAPTIVE SYSTEM (ANFIS)

In order to design a fuzzy model [2] we should make the following steps: outputs are associated to input member functions which are turned into laws in order to get a certain set of output features; the output feature changes into output member functions, and the latter turn into a single output value or a decision associated to the output. If we want to shape up such a system, based on an input-output data-set, we should use the ANFIS shaping [6].

During the first stage, we create the input-output data set which we are going to use during the training stage.

The next stage is to design a FIS structure which is going to be used in order to shape up a system which must acquire the input-output data pair.

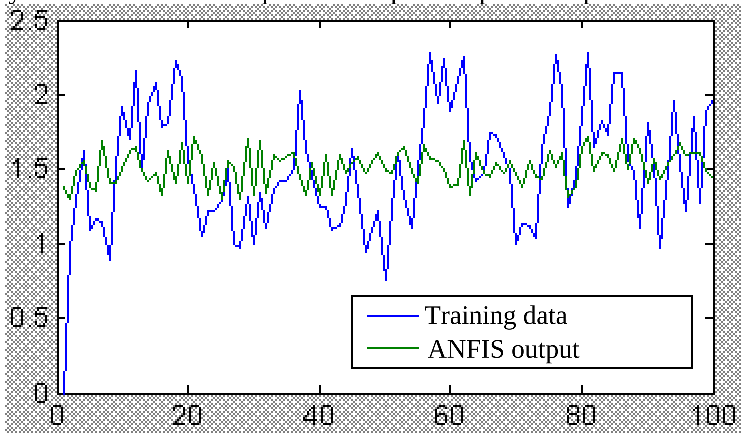
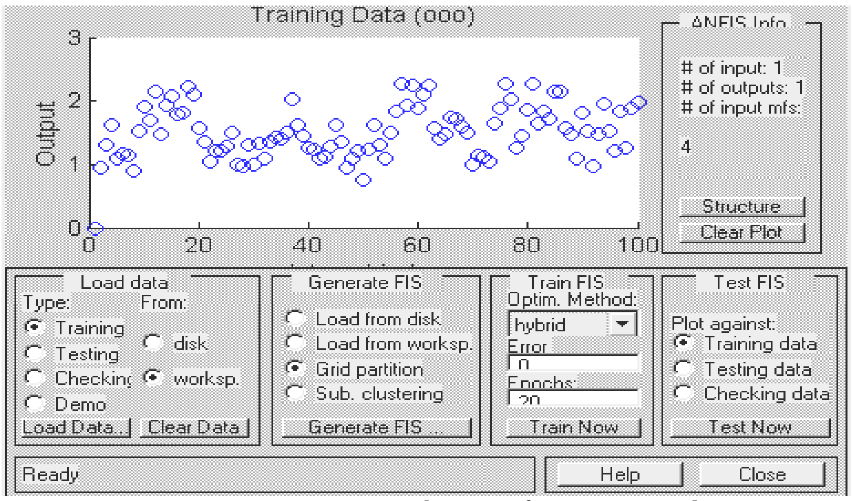


Figure no. 3 Process and network output

The FIS structure we have already created is going to be used in order to get a primary set of member functions used in training.

In order to check out the FIS system we have used, we should compare its output to the „y" input belonging to the training set. The output of the system we use and the „y" size we have obtained during experiments are described in Figure no. 3.

In order to identify [3] the ANFIS process, there is the possibility of using a graphical interface. Once we use this interface, it is possible to upload the data set in order to turn the network active – these data is uploaded on the work space or on the disk. After we have uploaded the data, the output data are graphically described alongside some information about them (number of entries, number of entry pairs) - Figure no. 4.



ANFIS Info
# of input: 1
# of outputs: 1
# of input mfs:
4

Load data
Type: [✓ Training] Testing Checking Demo
From: disk [✓ worksp.]

Generate FIS
Load from disk
Load from worksp.
[✓ Grid partition]
Sub. clustering

Train FIS
Optim. Method: hybrid
Error tolerance: 0
Epochs: 20

Test FIS
Plot against: [✓ Training data] Testing data Checking data

Ready

Figure no. 4 ANFIS graphic interface; training data

The next step is to define the FIS structure which is about to be used (Figure no. 5).

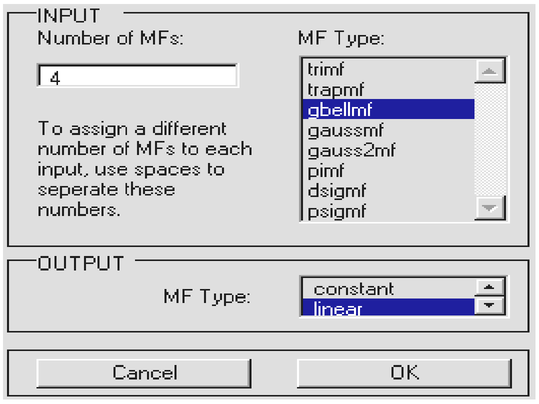

INPUT
Number of MFs:
4
To assign a different number of MFs to each input, use spaces to seperate these numbers.
MF Type:
trimf
trapmf
gbellmf
gaussmf
gauss2mf
pimf
dsigmf
psigmf
OUTPUT
MF Type:
constant
linear
Cancel
OK

Figure no. 5 Choosing the FIS network parameters

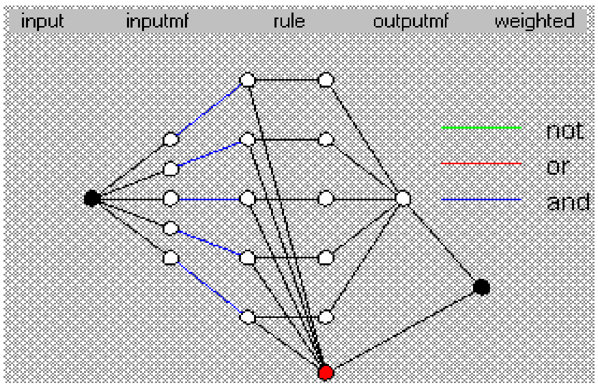


Figure no. 6 Network structure

The structure we have created could be graphically described. Therefore, the primary relations, the logical operations, and the implemented laws are ready to be performed - Figure no. 6.

After we have defined the parameters, we are able to perform the network training. The value of the error must be equal to 0; the best method of training is either '*hybrid*' or '*back propagation*' - the number of repeated stages to be performed. Figure no. 7 represents the evolution of the error during each stage of the training. If the training of the network is performed again then, it is going to work below the error value it had reached earlier. We see that once the number of periods increases, the error decreases, but not below to a certain value. That value is the limit. When it is reached, the number of repetitions does not cause a significant error decrease. If the number of repetitions increases, it reaches the „y" value of the output – Figure no. 8.

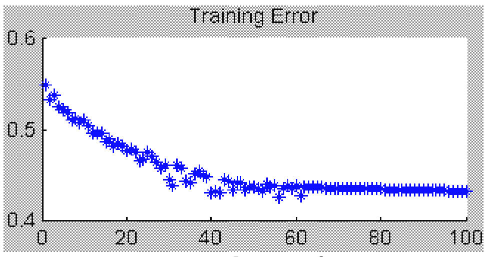


Figure no. 3.32 Evolution of ANFIS error

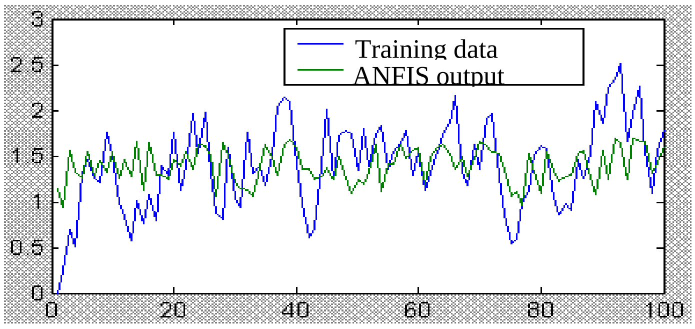


Figure no. 8 Evaluation of ANFIS after an increased number of repetitions

4. CONCLUSIONS

We have performed the shaping up of a system based on an input-output data set and we have used the ANFIS shaping-up method for that. We have created an input-output data set and we have created the FIS structure and the network training.

We have found out that once the number of periods increases, the error decreases, but it does not go below a certain margin. If it goes below that margin, the number of repetitions is increased and it does not produce any significant decrease of the error. If the number of repetitions increases, we find out that it reaches the right "y" value.

REFERENCES

[1] *Dumitrescu, D., Costin, H. „Reţele neuronale teorie şi aplicaţii" Editura Teora, 1996.*
[2] *Hong, X., Harris, C.J., Wilson, P.A., „Neurofuzzy state identification using prefiltering", IEE, 1999*
[3] *Ljung, L. "System Identification" Theory for the User Prentice Hall, Inc., Englewood Cliffs, New Jersey*
[4] *Narendra, K. S., Parthasaraty, K. "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Transactions on Neural Networks, vol. 1 nr. 1 1990, pag. 4-27.*
[5] *Tirian G.O. „High speed neuronal estimator for the command of the induction machine" International multidisciplinary scientific symposium"UNIVERSITARIA SIMPRO 2006" pg.62...65, 2006.*
[6] *G.O.Tirian „Abordari moderne in estimarea si identificarea sistemelor", Referat doctorat, Timisoara, 2005.*
[7] *Widrow, B., Lehr, M. A. "30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation", Proceedings of IEEE, vol. 78 nr. 9 1990, pag. 1415-1439*